INTERNATIONAL JOURNAL OF ADVANCED STUDIES
IN COMPUTER SCIENCE AND ENGINEERING
IJASCSE VOLUME 5 ISSUE 11, 2016

11/30/2016

The Performance Of Inner Join Types in SQL

Fawwaz Yousef Alnawaj'ha
IT Department, College of Applied Profession
Palestine Polytechnic University (PPU)
Hebron, Palestine
fawwaz@ppu.edu

***Abstract*—** The paper compared the performance of four of inner join types; NATURAL JOIN, JOIN … USING, JOIN … ON, and the traditional join or comma join, or WHERE clause join.

NATURAL JOIN ; it is used in joining two tables that have columns with the same name, JOIN … USING; is used if several columns share the same name but you don't want to join using all of these common columns. You can determine the common column you want to be used in join in the USING clause, JOIN … ON; is used when join column names are different [1,2].

Every type of these inner join types has its own conditions, but the question is: If there is a query that meet and comply the conditions of all of these types, which of them have better performance or which of them is the fastest?.

To answer the question we prepared a simple query in Oracle 10g to join Employees and Jobs tables, the query written in four ways to meet the four types of the inner join in SQL, the first by using NATURAL JOIN, the second by using JOIN … USING, the third by using JOIN … ON, and the forth by using WHERE Clause, each query executed 30 times and the execution time is recorded for each one, and the time average for each query is calculated we found that the average of JOIN …ON is 0.0050s, the average of WHERE Join is 0.0053s, The average of NATURAL JOIN is 0.0077s, the average of JOIN …USING is 0.0083s, we conclude that these types of join can be arranged from fastest to lowest speed as follows: JOIN…ON, WHERE Join, INNER NATURAL JOIN, and JOIN …USING.

Keywords-- query; performance; join; inner join; natural join.

I. INTRODUCTION

INNER JOIN is the most commonly and frequently used in joining tables in SQL, it refers also to EQIIJOIN [3], The INNER JOIN joins two or more tables, returning only the rows that satisfy the JOIN condition [4]. INNER JOIN returns a result table for all the rows in a table that have one or more matching rows in the other table(s), as specified by the sql-expression.

A two-table inner join may be viewed as the intersection between table A and table B as shown in the following Venn diagram in fig.1 . [7]

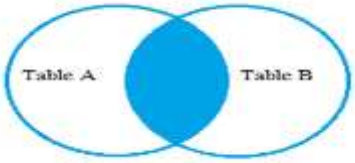

Fig.1: Venn Diagram of two joined tables

The fig.2 shows all types of SQL join including inner join [14]:

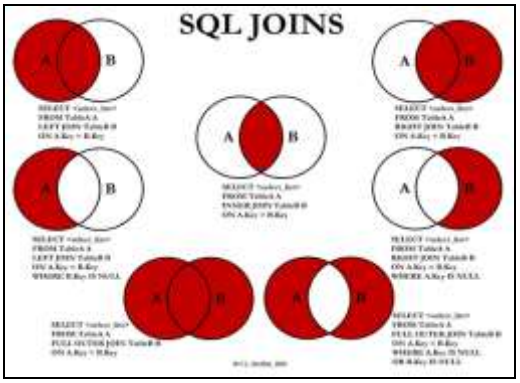

Figure 2: SQL Joins

II. TYPES OF INNER JOIN

There are four types of inner join; NATURAL JOIN, JOIN … USING, JOIN … ON, and WHERE clause join. the first three types of inner join are ANSI standard refers to SQL/99, the word INNER can be used optionally before the word JOIN. [1]

INNER JOIN can be specified either by explicit or implicit join condition clause as follows:

A. *Join By ON Clause*

INNER JOIN … ON ; Is the type that has an explicit join condition used when the join columns have different names [1,2], its basic syntax as follows:

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.field1 = table2.field2; [4]
```

Example:

```
SELECT D.Department_Name, E.Last_Name As
"Department Manger"
FROM Departments D INNER JOIN Employees E
ON ( D.Manager_ID = E.Employee_ID) ;
```

B. *Join By WHERE Clause*

WHERE clause, such as "WHERE table1.column1 = table2.column2" can be used to perform a join between two tables using an explicit join condition. [5]

The syntax:

```
SELECT table1.column1, table2.column2...
FROM table1, table2
WHERE table1.field1 = table2.field2;
```

Example:

```
SELECT D.Department_Name, E.Last_Name As
"Department Manger"
FROM Departments D, Employees E
WHERE  D.Manager_ID = E.Employee_ID ;
```

C. *Join By NATURAL Clause*

A NATURAL JOIN is a JOIN operation that creates an implicit join clause based on the common columns in the two tables being joined, common columns are columns that have the same name in both tables. [5]

The syntax:

```
SELECT table1.column1, table2.column2...
FROM table1 NATURAL JOIN  table2;
```

Example:

```
SELECT E.Last_Name, D.Department_Name
FROM Departments D NATURAL JOIN Employees E;
```

D. *Join By USING Clause*

JOIN … USING; The USING clause specifies which columns to test for equality when two tables are joined. It can be used instead of an ON clause in the JOIN operations that have an explicit join clause when the two tables have more than one pairs of a common columns.[5]

The syntax:

```
SELECT table1.column1, table2.column2...
FROM table1  JOIN  table2
USING (common_column1, common_column2, … );
```

Examples:

The following query performs an inner join between the COUNTRIES table and the CITIES table on the condition that COUNTRIES.COUNTRY is equal to CITIES.COUNTRY:

```
SELECT * FROM COUNTRIES JOIN CITIES
 USING (COUNTRY);   [6]
```

The next query is similar to the one above, but it has the additional join condition that COUNTRIES.COUNTRY_ISO_CODE is equal to CITIES.COUNTRY_ISO_CODE:

```
SELECT * FROM COUNTRIES JOIN CITIES
 USING (COUNTRY, COUNTRY_ISO_CODE);  [6]
```

When execute an SQL query in Oracle, Oracle creates a query execution plan in order to retrieve data from the tables. The query execution plan is a set of methods on how the database will access the data from the tables. The query may have different execution plans that need different cost and time for the query execution, the query optimizer chooses the more efficient plan for data retrieval to achieve better performance that means fast response on data retrieval action .[9]

III. Research Problem/Question

Every type of these inner join types has its own conditions, but the question is: If there is a query that meet and comply the conditions of all of these types, which of them have better performance or which of them is the fastest?.

IV. Objectives

The specific objectives of this paper are:

- To shed light on the types of the inner join in SQL.
- To study the performance of inner join types by comparing the averages of the execution time of each type.

V. Previous Studies

There are a lot of scientific papers conducted in the area of query performance including join tables and the types of join , and here are some of these papers that the researcher has the opportunity to look on :

- Synametrics Technologies Inc. 2014; They published a paper that listed 10 tips that every developer/DBA should consider when designing their database or writing SQL scripts. It also talks about some common mistakes developers typically make and how to avoid them.[11]
- Hristo Kyurkchiev, Kalinka Kaloyanova 2014; they developed a benchmark, which is subsequently used to

measure some of DBMS's performance. Evaluating the results then decide about each DBMS's suitability and main advantages over the other. [12]

- Ossama K. Muslih, Imad Hasan Saleh 210; They are trying in their paper to enhance the performance of the database by sending a well done, error free and a professional SQL commands, They proposed a system that will try to find any SQL statement with join predicate and examine if the statement has non join predicate, the system will rewrite the statement and send a hint to database optimizer to start with non join predicate. [16]
- Dalia C. Kahane 2008; demonstrates in paper the importance of knowing your data and the fields that are common to all datasets or unique in each; and the syntax for properly performing different types of joins in SQL (inner vs. outer join, left vs. right join, etc.) [7]
- Kerry Osborne's 2008, found the Cost Based Optimizer in Oracle is sometimes erratic performance, and he shows how the plans for a given statement have changed internally over time, along with some statistics such as the average elapsed time and the average amounts of logical input/output. [8]
- Ben Nadel 2006, he Said that " I have been under the impression that putting 'criteria' in the ON clauses of a SQL statement will create a faster query when compared to the same SELECT statement if the same criteria was placed in the WHERE clause. " [14]
- Roger Schrag 2005, In his paper he discussed the semi-join and the anti-join, two powerful SQL constructs Oracle offers for use in quest for faster queries. In particular, He defined these two terms, discussed when and why you might want to use the [NOT] EXISTS or [NOT] IN constructs, and demonstrate how can use optimizer hints and make minor query changes in order to enable Oracle to use some very powerful and efficient access paths. [10]

VI. Research Significance

In database SQL query performance becomes an issue sooner or later. Having long-running queries not only consumes system resources that makes the server and application run slowly, but also may lead to table locking and data corruption issues. So, query optimization becomes an important task.[13]
Most of queries using join between tables, The join process in relational databases are important for data collection from more than one table in order the retrieved data to be consistent, complete, and integrated.

The fact that there are different types of join tables specially in inner join, it is significant for developers of database systems to understand these types and know which of them has better performance to be used in building a good queries that serve their purpose in lowest cost.

VII. Methodology

A. Procedures

"Performance Improvement means doing things faster", "Performance is always and only about time" [17]. The query performance method is rooted in finding performance problems that can be seen by using statistics, so we have to collect some statistics about an important factor that affect the query performance, that is the time response or the execution time of the query, to do that a simple query prepared in Oracle 10g to join Employees and Jobs tables, the query written in four ways to meet the four types of the inner join conditions in SQL, the first by using NATURAL JOIN, the second by using JOIN … USING, the third by using JOIN … ON, and the forth by using WHERE Clause, each query executed 30 times and the response time of each query is recorded and the average time is calculated for each query. These queries as follows:

The first query:

```
SELECT last_name, job_title
FROM  Employees NATURAL JOIN  Jobs ;
```

The second query:

```
SELECT last_name, job_title
FROM Employees JOIN  Jobs USING(job_id) ;
```

The third query:

```
SELECT last_name, job_title
FROM  Employees E   JOIN  Jobs J   ON(E.job_id=J.job_id)
;
```

The forth query:

```
SELECT last_name, job_title
FROM  Employees E , Jobs J
WHERE  E.job_id=J.job_id ;
```

Each query executed 30 times and the execution time is recorded for each one, and the time average for each query is calculated as in the table_1:

Table 1: Time average of query execution in each type of join

Join Type	Natural Join	Join Using	Join On	Where
Time Average (second)	0.0077	0.0083	0.0050	0.0053

B. The Result

As a table 1 and the histogram in fig.3 shows that the speed average of NATURAL JOIN, JOIN … USING, JOIN…ON, and WHERE Join are respectively: 0.0077 sec., 0.0083 sec., 0.0050 sec., and 0.0053 sec., hence these types of inner join can be arranged from fastest to lowest speed as follows: JOIN…ON, WHERE Join, NATURAL JOIN, and JOIN … USING.

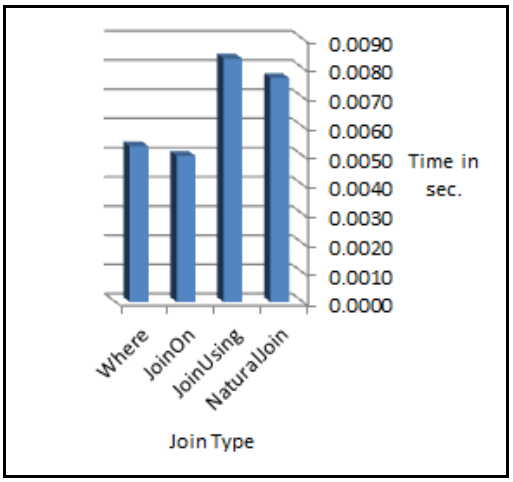

Fig.3: Histogram between join type and time average of execution

When referring to the query plan of each query in fig.4 we found that these query executed with same plan, with same cost even though the difference in speed of execution expressed in the histogram, this result may agree with what said by Kerry Osborne's when he concluded that "One of the most frustrating things about the Cost Based Optimizer in Oracle is the sometimes erratic performance. It can seem almost random at times." [8], Because oracle 10g using Cost Based Optimizer and the queries that we talks about is executed in oracle 10g, same result reached by Ben Nadel when he tested ON clause vs WHERE clause and the result was; ON clause created a faster query than WHERE clause [14].


Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes
SELECT STATEMENT			107	1	4	4,708
NESTED LOOPS			107	1	4	4,708
TABLE ACCESS	FULL	EMPLOYEES	107	1	3	1,819
TABLE ACCESS	BY INDEX ROWID	JOBS	1	1	1	27
INDEX	UNIQUE SCAN	JOB_ID_PK	1	1	0	

Fig.4: Query plan of query execution

VIII. CONCLUSION

This paper explained the concept of inner join and its types, explore the performance of each type of inner join, compared these types in term of speed, we conclude that these types of join can be arranged from fastest to lowest with related to execution speed as follows: JOIN…ON, WHERE Join, INNER NATURAL JOIN, and JOIN …USING.

IX. RECOMMENDATIUONS

- We recommend doing more studies on the execution speed of these types of the inner join, to make sure of these results.
- according to the result achieved we recommend SQL developers to adopt inner join types with explicit join conditions in their queries like using: JOIN .. ON and WHERE clause, because these types of join compared to the other types has less execution time.
- The paper raised another issue for farther work , that is worthy to study; the issue is the exact reasons led to the differences in execution time (response time) between the inner join types.

REFERENCES

[1] B.Consulting, "Oracle simplifies SQL with ISO 99 Syntax", http://www.dba-racle.com/art_sql_iso_99.htm, 2002, accessed in 6/10/2015.
[2] R.Gravenstein, "To ANSI or Not To ANSI Session #: 420", csit.parkland.edu, accessed in 6/10/2015.
[3] TutorialsPoint , "SQL - INNER JOINS", http://www.tutorialspoint.com/sql/sql-inner-joins.htm , visited in 9/10/2015.
[4] WIKIBOOKS, "Oracle Programming /10g Advanced SQL" , https://en.wikibooks.org/wiki/Oracle_Programming/10g_Advanced_SQL, accessed in 9/10/2015.
[5] Oracle Help Center, "JOIN Operations" , https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj29840.html, accessed in 9/10/2015.
[6] Oracle Help Center, "USING Claus"e, http://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljusing.html , accessed in 10/10/2015.
[7] D. C. Kahane and others, "Using DATA Step MERGE and PROC SQL JOIN to Combine SAS® Datasets", NESUG 2008 , http://www.lexjansen.com/nesug/nesug08/ff/ff03.pdf.

[8] K. Osborne's, "Unstable Plans (Oracle Plan Stability/Instability)", 8 October 2008, http://kerryosborne.oracle-guy.com/2008/10/unstable-plans/

[9] A. Mitra, "Understanding Oracle QUERY PLAN", 17 Tune 2014, http://dwbi.org/database/oracle/38-oracle-query-plan-a-10-minutes-guide

[10] R. Schrag ," Speeding Up Queries with Semi-Joins and Anti-Joins: How Oracle Evaluates EXISTS, NOT EXISTS, IN, and NOT IN", 2005 Database Specialists Inc., http://www.dbspecialists.com/files/presentations/semijoins.html

[11] Synametrics Technologies Incorporation, "Top 10 performance tuning tips for relational databases", 2013-2014 Synametrics Technologies, http://web.synametrics.com/top10performancetips.htm , accessed in 27/10/2015.

[12] H. Kyurkchiev and others, "Performance Study of Analytical Queries of Oracle and Vertica", Conference: Information Systems & Grid Technologies, At Sofia , Sofia, Bulgaria 2014.

[13] 1Keydata, "Query Optimization" , http://www.1keydata.com/datawarehousing/query-optimization.html , accessed in 27/10/2015.

[14] Stack Exchange, "Database Administrator", http://dba.stackexchange.com/questions/73087/mysql-which-join-is-better-between-left-outer-join-and-inner-join , accessed in 30/10/2015.

[15] B. Nadel, "SQL Optimization And ON Clause vs WHERE Clause" , http://www.bennadel.com/blog/284-sql-optimization-and-on-clause-vs-where-clause.htm , Published 2006, accessed in 30/10/2015.

[16] O. Muslih and others, "Increasing Database Performance through Optimizing Structure Query Language Join Statement " , Journal of Computer Science 6 (5): 585-590, 2010, ISSN 1549-3636.

[17] G. Wood, "Database Time_Based Performance Tuning From Theory to Practice" , Oracle University 2014, https://www.youtube.com/watch?v=Aknb_iZrPbc, accessed in 4/1/2016.